Mathematics for Programmers

This article will talk about the role of mathematics in the life of a software developer. We will not delve into specific areas like machine learning, modeling or computer graphics but will focus on basic math stuff.

This material is meant primarily for those who have already taken their first steps in the IT industry but have spent more time in their education on programming languages and specific technologies rather than on the fundamentals.

How to study math

Many people find math to be a very difficult science to understand. This is often because they don't approach it in the right way. The truth is that you can simplify your life a lot by following the recommendations below.

There are two levels of understanding of mathematics. The first level is conceptual. This is an awareness of what certain objects are for, what problem is solved, and where it is used. The second level of understanding is detailed; it is a detailed study of the details of solving a problem. Sometimes you need to understand the task at the detailed level of understanding, but in most cases, the ideational level is enough.

Math doesn't like buzzwords. If you are reading a book and see words whose meaning you don't understand, skipping them is dangerous, because you might catch yourself at some point not understanding anything at all. It's very important to stop yourself right away when you don't understand something.

Discrete Mathematics.

The field of mathematics which deals with discrete structures (e.g.: graphs, automata, statements in logic). Its main difference from the regular math you learned in school is that its objects can't change as smoothly as real numbers.

In a sense, all of the problems you solve in programming relate to discrete mathematics in one way or another, so knowing it will come in handy for you.

Logic

This is the science of formal systems and proofs. It is the basis of computer science, because any programming language is a formal system. But you don't have to look too deeply into the theory to apply it to programming and solving problems in general.

It is good if you know how to write a solution to a problem. But it's also important to understand how you can prove that your code works correctly. Most programs solve a mathematical problem, and you need to be able to prove that your problem was solved correctly. That's where the methods of logic and, in particular, the calculus of statements come in.

It is a good idea to start with simple things: for example, what an utterance is, what are the operations between them, and what are the rules of inference. Then you can move on to more applied areas: try to solve logic problems, try to optimize the various checks that you have to write in code. Next, you should pay attention to first-order logic: it can be useful in testing programs.

However, the solution that came to your mind first is not always the most correct and beautiful. Often, formal conversions can reduce the amount of code and make it more readable. And in addition, some logical tricks can make the solution itself shorter, faster, and more efficient.

Resources:

On Codeforces in the Archive section, it's worth practicing on at least C-class problems-many of them contain pitfalls;

The KeY to Software Correctness project is good for those who are curious about how you can automatically prove code is correct. It automates code verification in Java;

Automatic theorem prover z3, written by Microsoft, for those who use other languages. A short instruction on how to use it can be found at rise4fun.

Combinatorics

Combinatorics studies various discrete sets and the relations of their elements. The most common combinatorial problem encountered by programmers is to derive the number of elements that need to be enumerated to get a solution depending on some parameters. In this way you can derive the asymptotic complexity of the algorithm.

Combinatorial problems are formulated as a problem of counting the number of elements of some (in mathematics the term power is used) set. To solve such problems it is useful to have basic knowledge in set theory from the category of properties of operations on sets. The problem then boils down to expressing the sought set in sets, powers of which are calculated according to known rules. The rules of multiplication or addition, numbers of combinations or placements are used to

calculate the number of elements. Although there are more complicated problems, it is best to start with simple ones.

Programming Help Online

Resources:

The basics can be found at Mathprofi, a website dedicated to transparent and popular descriptions of mathematics;

If you speak English, you can check out the more advanced book An Introduction to Combinatorics and Graph Theory;

Combinatorics problems can be taken from the Discrete Mathematics problem book; there are answers at the end.